

PATENT APPLICATION

NETWORK CACHE-BASED CONTENT ROUTING

Inventor(s): Martin Cieslak of
Fremont, California
United States citizen

James Aviani of
Santa Barbara, California
United States citizen

Martin Kagan of
Burlingame, California
United States citizen

Assignee: Cisco Technology Inc.

BEYER WEAVER & THOMAS, LLP
P.O. Box #130
Mountain View, CA 94042-0130
(510) 843-6200

03569027.00000000

PATENT

Attorney Docket No. CISCP139

NETWORK CACHE-BASED CONTENT ROUTING

BACKGROUND OF THE INVENTION

The present invention relates to transmission of data in a network environment.

More specifically, the present invention relates to methods and apparatus for improving the efficiency with which data are transmitted over the Internet. Still more specifically, the present invention provides techniques by which routing decisions for particular data packets are made with reference to the nature of the data being transmitted, e.g., content routing.

Generally speaking, when a client platform communicates with some remote server, whether via the Internet or an intranet, it crafts a data packet which defines a TCP connection between the two hosts, i.e., the client platform and the destination server. More specifically, the data packet has headers which include the destination IP address, the destination port, the source IP address, the source port, and the protocol type. The destination IP address might be the address of a well known World Wide Web (WWW) search engine such as, for example, Yahoo, in which case, the protocol would be TCP and the destination port would be port 80, a well known port for http and the WWW. The source IP address would, of course, be the IP address for the client platform and the source port would be one of the TCP ports selected by the client. These five pieces of information define the TCP connection.

Given the increase of traffic on the World Wide Web and the growing bandwidth demands of ever more sophisticated multimedia content, there has been constant pressure to find more efficient ways to service data requests than opening direct TCP connections between a requesting client and the primary repository for the desired data. Interestingly,

one technique for increasing the efficiency with which data requests are serviced came about as the result of the development of network firewalls in response to security concerns. In the early development of such security measures, proxy servers were employed as firewalls to protect networks and their client machines from corruption by undesirable content and

5 unauthorized access from the outside world. Proxy servers were originally based on Unix machines because that was the prevalent technology at the time. This model was generalized with the advent of SOCKS which was essentially a daemon on a Unix machine. Software on a client platform on the network protected by the firewall was specially configured to communicate with the resident demon which then made the connection to a destination

10 platform at the client's request. The demon then passed information back and forth between the client and destination platforms acting as an intermediary or "proxy."

Not only did this model provide the desired protection for the client's network, it gave the entire network the IP address of the proxy server, therefore simplifying the problem of addressing of data packets to an increasing number of users. Moreover, because of the storage capability of the proxy server, information retrieved from remote servers could be stored rather than simply passed through to the requesting platform. This storage capability was quickly recognized as a means by which access to the World Wide Web could be accelerated. That is, by storing frequently requested data, subsequent requests for the same data could be serviced without having to retrieve the requested data from its original remote

15 source. Currently, most Internet service providers (ISPs) accelerate access to their web sites using proxy servers.

Unfortunately, interaction with such proxy servers is not transparent, requiring each end user to select the appropriate proxy configuration in his or her browser to allow the browser to communicate with the proxy server. For the large ISPs with millions of

20 customers there is significant overhead associated with handling tech support calls from

customers who have no idea what a proxy configuration is. Additional overhead is associated with the fact that different proxy configurations must be provided for different customer operating systems. The considerable economic expense represented by this overhead offsets the benefits derived from providing accelerated access to the World Wide Web. Another problem arises as the number of WWW users increases. That is, as the number of customers for each ISP increases, the number of proxy servers required to service the growing customer base also increases. This, in turn, presents the problem of allocating packet traffic among multiple proxy servers.

Another technique for increasing the efficiency with which data requests are serviced is described in commonly assigned, copending U.S. Patent Application No. 08/946,867 for METHOD AND APPARATUS FOR FACILITATING NETWORK DATA TRANSMISSIONS filed October 8, 1997, the entirety of which is incorporated herein by reference for all purposes. The invention described in that copending application represents an improvement over the proxy server model which is transparent to end users, high performance, and fault tolerant. By altering the operating system code of an existing router, the router is enabled to redirect data traffic of a particular protocol intended for a specified port, e.g., TCP with port 80, to one or more caching engines connected to the router via an interface having sufficient bandwidth such as, for example, a 100baseT interface. If there are multiple caching engines connected to the cache-enabled router, the router selects from among the available caching engines for a particular request based on a simple algorithm according to which a particular group of addresses is associated with each caching engine.

The caching engine to which the request is re-routed "spoofs" the requested destination platform and accepts the request on its behalf via a standard TCP connection established by the cache-enable router. If the requested information is already stored in the caching engine, i.e., a cache "hit" occurs, it is transmitted to the requesting platform with a

header indicating its source as the destination platform. If the requested information is not in the caching engine, i.e., a cache "miss" occurs, the caching engine opens a direct TCP connection with the destination platform, downloads the information, stores it for future use, and transmits it to the requesting platform. All of this is transparent to the user at the requesting platform which operates exactly as if it were communicating with the destination platform. Thus, the need for configuring the requesting platform to suit a particular proxy configuration is eliminated along with the associated overhead. Moreover, traffic may be easily allocated among as many caching engines as become necessary. Thus, content caching provides a way to compensate for the bandwidth limitations discussed above.

Another specific embodiment of a packet redirection protocol which may be used to implement such a network caching technique is described in copending, commonly assigned, U.S. Provisional Patent Application No. 60/168,862 for METHOD AND APPARATUS FOR REDIRECTING NETWORK TRAFFIC filed December 2, 1999, the entirety of which is incorporated herein by reference for all purposes. According to a specific embodiment described in that application, the network caches have the capability of determining that particular redirected packets should be transmitted back to the redirecting router and reinserted into the original traffic flow. This may be done in a manner transparent to the source or destination of the packets. An example of a case in which packets would need to be reinserted in the original flow might be where the cache recognizes the source and destination pairs identified by the packets as corresponding to a connection requiring IP-based authentication. Another example would be where the cache is overloaded and is currently unable to handle all of the redirected traffic.

A technique conceptually related to content caching referred to as content co-location has also been developed to provide faster access to certain types of data. That is, a large amount of Internet content resides on machines in the U.S. Client machines in remote

geographic locations such as New Zealand or Australia typically have two ways of retrieving such content, i.e., via terrestrial links (bidirectional and fast, but expensive) and satellite links (mostly unidirectional and relatively cheap, but slow).

Content co-location provides one way to facilitate faster access to such content for geographically remote users. According to this technique, servers are provided in the geographically remote areas which store at least some of the content of the original site, e.g., image files. Akamai and Sandpiper provide such “co-loc” facilities to content providers.

When a client requests a page containing such objects, the html refers to the objects by object names which correspond to the co-loc provider. These object names are then resolved by the client’s browser to an IP address corresponding to the closest co-loc facility.

While content co-location techniques have proven to be relatively effective, content providers pay a huge premium for this service. In addition, implementation of content co-location requires some relatively complex and computationally intense protocols. For example, coherency must be maintained between content at co-loc facilities and the content at the original site. In addition, browsers must constantly perform the object name resolution when parsing ordinary html to access the correct co-location facility.

Another limitation associated with various network devices, e.g., routers, switches, gateways, and some caches, is that routing decisions are made only with reference to the lower level network layers, i.e., the physical, data link, and network layers. Fig. 1 illustrates the relationship of one such network device, i.e., a gateway, to the various network layers.

Assume that the user application layer protocol 102A in host A sends an application protocol data unit (PDU) to an application layer protocol 102B in host B, such as, for example, a file transfer system. The file transfer software performs a variety of functions and sends file records to the user. In many systems, the operations at host B are known as *server* operations and the operations at host A are known as *client* operations.

As indicated by the downward arrows in the protocol stack at host A, this unit is passed to the transport layer protocol 104A, which performs a variety of operations and adds a header to the PDU passed to it. At this point, the unit of data is often referred to as a *segment*. The PDU from the upper layers is considered to be data to the transport layer.

5 Next, the transport layer passes the segment to the network layer 106A, also called the IP *layer*, which again performs specific services and appends a header. This unit (now called a *datagram* in internet terms) is passed down to the lower layers. Here, the data link layer adds its header as well as a trailer, and the data unit (now called a *frame*) is launched
10 into subnetwork 110 by the physical layer 108A. Of course, if host B sends data to host A, the process is reversed and the direction of the arrows is changed.

15 Internet protocols are typically unaware of what goes on inside the network. The network manager is free to manipulate and manage the PDU in any manner necessary. In some instances, however, the internet PDU (data and headers) remains unchanged as it is transmitted through the subnet. In Fig. 1, it emerges at the gateway where it is processed through the lower layers 114 and passed to the IP (network) layer 112. Here, routing decisions are made based only on the destination address provided by the host computer.
20 After these routing decisions have been made, the PDU is passed to the communications link connected to the appropriate subnetwork (comprising the lower layers). The PDU is re-encapsulated into the data link layer frame and passed to the next subnetwork 116, where it finally arrives at the destination host.

One of the limitations inherent in the traffic routing approach described above is that the routing decisions are made with reference only to information in the lower network layers. Because data transmissions in these layers are virtually indistinguishable from one another it is difficult if not impossible to make intelligent routing decisions based on the type

of data being transmitted or requested, or any other high level information. As a result, opportunities for improving the efficiency with which networks operate are lost.

In view of the foregoing, there is a need for techniques by which network devices can make intelligent routing decisions about specific types of data traffic which are less expensive in terms of network resources than currently available solutions.

0552022 - 059500

SUMMARY OF THE INVENTION

According to the present invention, network devices are enabled to make intelligent routing decisions based upon any of a number of parameters or characteristics associated with specific data traffic. According to a specific embodiment of the invention, the 5 parameters or characteristics of interest are associated with the application layer. According to another specific embodiment, the parameters or characteristics of interest relate to information which is specific to a network cache making the routing decisions.

Thus, the present invention provides a computer-implemented method for routing data traffic in a network having a plurality of layers including physical, data link, and 10 network layers. The data traffic is received and one of a plurality of routing options for the data traffic is selected with reference to information outside of the physical, data link, and network layers. The data traffic is then routed according to the selected routing option.

According to one embodiment, the data traffic is received and one of a plurality of routing options for the data traffic is selected with reference to information associated with 15 the network's application layer. In another embodiment where the data traffic has been redirected to a network cache, the data traffic are received by the network cache which selects one of a plurality of routing options for the data traffic with reference to information about the data traffic accessible by the network cache.

A further understanding of the nature and advantages of the present invention may be 20 realized by reference to the remaining portions of the specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram illustrating conventional data transmissions in a wide area network;

Fig. 2 is a block diagram of a network environment in which specific embodiments of the present invention may be implemented;

Fig. 3 is a block diagram of a network cache for use with specific embodiments of the present invention; and

Fig. 4 is a flowchart illustrating the caching of objects according to a specific embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

Fig. 2 is a block diagram of a network environment in which specific embodiments of the present invention may be implemented. According to a first specific embodiment, a request for content in the United States is received by a network cache 210 in Australia.

5 Based on information associated with the application layer, cache 210 decides whether to use a terrestrial link or a satellite link to route the request. The selected link will thus determine the routing path for the response to the request. According to one embodiment, the determination is made by parsing the request in the application layer and then selecting a socket option for the outgoing request accordingly. In the figure, the terrestrial link is shown

10 comprising router 212, a wide area network 214 of arbitrary size, and router 216. The satellite link is shown comprising router 218, wide area network 220, satellite link 222, satellite 224, satellite link 226, wide area network 228, and router 230. Each of the links is shown connecting with a cache 232 although this is not necessary for implementation of this embodiment. That is, the links may connect directly with the destination machine or an

15 associated router.

According to a more specific embodiment, the decision between the terrestrial and satellite links is based on information which correlates to the relative sizes of the objects being requested. That is, html objects are typically fairly small as compared to graphically intense embedded objects. In addition, html objects typically anchor or form the foundation for web pages. When a user is downloading such a page, they will tolerate a slower download of the graphics, but they need the page's foundation, i.e., the html, right away. Therefore, according to this embodiment, the html is requested via the fast terrestrial link and the embedded graphics via the slower satellite link. According to a still more specific embodiment, the type of object is determined (i.e., the routing decision is made) with reference to the suffix of the URL (e.g., *.htm vs. *.gif).

The embodiments described above use the satellite uplink to request the larger objects. That is, because the decision making is performed on the side of the links which is remote from the requested content, satellite link 222 must be used to both transmit the request and receive the requested data. Because providing this uplink capability at satellite 5 link 222 can be expensive, a second embodiment is provided in which cache 232 on the content side of the links makes the decisions. That is, when cache 232 receives data traffic, it routes the traffic using, for example, the criteria described above with reference to cache 210. The difference is that, because cache 232 is on the content side of the links, no satellite 10 uplink needs to be provided. Thus, for example, as with the embodiment described above, because of the application layer characteristics of data traffic received by cache 232, large objects (e.g., *.gif objects) may be sent via the satellite link while small objects (*.htm objects) may be sent via the terrestrial link.

It will be understood that the network configurations and links shown in Fig. 2 are merely used for illustration purposes and that a wide variety of configurations may be 15 employed without departing from the scope of the invention. In addition, the routing decisions enabled by the present invention are not limited to a choice between a terrestrial link and a satellite link, or based only on application layer information as described above. Rather, the present invention enables intelligent selection from among a plurality of routing 20 options based on any combination of information associated with the application layer and any of the other network layers. Put another way, based upon the request, the cheapest of a plurality of links is selected for transmitting the request and thus the object. The determination of what is cheapest is a multi-parameter determination which looks at, for example, speed, economic cost, specific user's need/level of access, remote ISP use 25 agreements (e.g., many people overseas have multiple ISPs for which the cost of use increases dramatically at certain levels thereby making it reasonable to switch over to

another ISP), etc. Effectively, the present invention enables so called content routing, i.e., the notion that objects are routed based on the type of content requested.

The information upon which the routing determination is based may vary widely. That is, network caches designed according to the present invention may incorporate routing rules based on any information associated with the higher network layers. For example, a request or a requested object may be routed according to whether the requested content is cacheable or uncacheable, ascii or binary, HTTP or ICP, regular requests or forced reloads, static web page requests or browser-based applications. For example, an object may be determined to be cacheable by looking at the HTTP headers or URL suffixes. In addition, whether an object is ascii or binary may be determined with reference to specific file suffixes, e.g., .htm vs. .gif or .jpg. An object can be differentiated as HTTP or ICP with reference to the transmission protocol used (i.e., TCP vs. UDP) or the port. A regular request may be distinguished from a forced reload with reference to the appropriate HTTP header.

Fig. 3 is a block diagram of a network cache such as, for example, cache 210 or cache 232 of Fig. 2. A central processor 302 controls operation of cache 210/232 and its various subsystems using system memory 304 and bus 306. Data objects are stored in cache memory 308 which, in a specific embodiment, comprises multiple queues of volatile RAM 310. According to various embodiments, memory 308 may comprise one or more nonvolatile disk drives. According to yet other embodiments, memory 308 may comprise any combination of volatile and nonvolatile memory. According to a specific embodiment, a nonvolatile disk drive 311 is provided as additional storage for cached objects.

A network interface 312 enables communication with external devices. Portions of memory 310 may also be employed for other purposes such as, for example, storing software code for directing the operation of various functionalities of cache 210/232. Alternatively,

program instructions for execution by processor 302 directing operation of the functionalities of cache 210/232 may be stored in a separate program memory 305. It will be understood that the cache architecture shown in Fig. 3 is merely illustrative and should not be construed to limit the scope of the present invention. That is, any of a wide variety of cache architectures may be employed to implement the present invention.

Network caches are in a unique position to have certain information about specific data traffic which is unavailable to other network devices. Some of this information is useful with regard to making intelligent routing decisions for the associated traffic.

According to a specific embodiment of the present invention, when a network cache receives a redirected request for an object from a cache-enabled router, the cache identifies the redirecting router from the encapsulation header and, if the cache needs to issue its own request for that object, it can route the new request through the redirecting router to avoid disruption of an independent load balancing scheme. This is important in caching systems in which multiple routers are logically connected to multiple caches.

Fig. 4 is a flowchart illustrating the routing of data traffic according to a specific embodiment of the present invention. When a packet is redirected to a cache designed according to the present invention (402), information associated with the packet is parsed (404) to determine which of a plurality of routing options are to be selected (406). The packet is then routed according to the selected routing option (408). According to a specific embodiment and as described above, the information associated with the packet is application layer information. According to yet other embodiments, cache specific information associated with particular data traffic is parsed for the routing determination.

While the invention has been particularly shown and described with reference to specific embodiments thereof, it will be understood by those skilled in the art that changes in the form and details of the disclosed embodiments may be made without departing from the

spirit or scope of the invention. Therefore, the scope of the invention should be determined with reference to the appended claims.